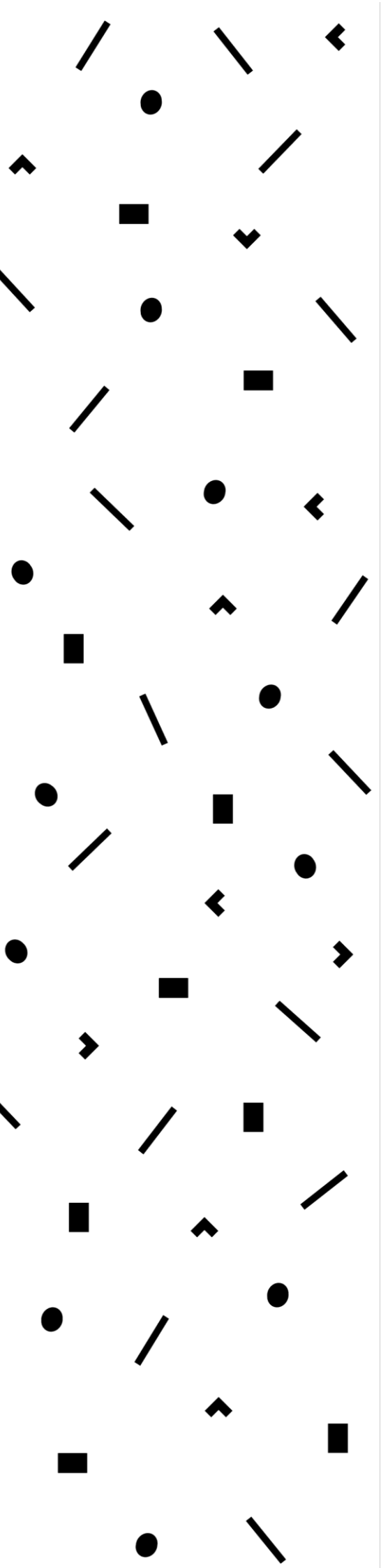

LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

HADOOP PERFORMANCE EVALUATION IN CLUSTER ENVIRONMENT

LAHTI UNIVERSITY OF APPLIED SCIENCES
Master's Degree Programme in Information and Communications Technology
Master´s Thesis
Autumn 2017
Fitsum Belay

Lahti University of Applied Sciences
Degree Programme in Information and Communications Technology

BELAY, FITSUM: Hadoop Performance Evaluation in Cluster Environment

Master's Thesis in ICT, 46 pages, 3 pages of appendices

Autumn 2017

ABSTRACT

With the growth of the internet a huge amount of data is being roduced every second. Companies rely on data analytics to expand their business and to stay competitive in the market. Over time the technologies of big data analytics have become more affordable for small companies.Unfortunately,small companies usually find it difficult to make the best use of the resources due to wrong assumptions about big data or because they are unable to meet the infrastructural requirements big data analysis involves.

There is a general assumption that big data is only for big businesses, which is not true. Companies usually unable to use the existing infrastructure to implement big data analytics and consequently fail to use an opportunity for growth. The purpose of this study was to encourage small companies to consider big data in their expansion strategies by showing them how big data analytics assists business, using the existing infrastructure.

One of the objectives of this thesis was to evaluate the performance of Hadoop cluster interms of input-output (I/O). This test gives a preliminary idea of how fast the cluster performs in terms of I/O and data throughput. The performance can be measured by feeding different sizes of data sets and changing the number of datanodes in the cluster. Throughout the whole process, Hadoop core components and were investigated.

According to the results, the performance of a multi node cluster in terms of average throughput is better than that of a single node Hadoop It can be concluded that even with an inexpensive infrastructure, by optimizing the existing resources, it is possible to process large volumes of data.

There are different factors that affect the performance of a cluster. These factors include the number of the files the cluster deals with and the processing power of the nodes. However, the network and hardware factors that might degrade the performance were not considered in this thesis.

.

Key words: Hadoop, TestDFSIO, stand-alone mode, pseudo-distributed mode, big data

CONTENTS

1 INTRODUCTION

Relational database is a collection of data items organized as a set of formally described tables from which data can be accessed or rearranged in a variety of ways without having to reorganize the database tables (TechTarget 2006).

Social media and other sources produce a huge amount amount of data every day, every second. Blogs and user forums contain potentially useful information. Relational databases can no longer handle such huge amounts of data. Due to this, a better approach is required to handle the large dataset and add extra value to use the data in a better way. Big data posesses three unique properties which are the velocity, variety and volume. Velocity refers to the speed of data accumulation while variety is the different formats of data available. Volume is the amount of data increase every second of time. The extra value from big data can be provided by implementing big data analytics. The Apache Hadoop open source alternative can help here (Elgendy & Elragal 2014).

The evolution of analytics has been made possible due to different major innovations. Improvements and advances in storage and processioning capabilities allow making use of big data sets that we have not been able to use before (Marr 2015).

1.1 Big data and Hadoop

The first documented use of the term “big data” appeared in a 1997 (NASA) problem with computer graphics. “Big data provides an interesting challenge for computer systems: data sets are generally large, taxing the capacity of the main memory, local disk and even remote disk. “

Apache Hadoop is an open source software platform for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. Hadoop services provide for data

storage, data processing, data access, data governance, security and operations. (Horntonworks 2017.)

Hadoop performance in terms of storage and analytics is a key issue for companies engaging in data centric business.

Apache Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage (Apache Software Foundation 2014).

1.2 Purpose of the thesis

The main aim of the thesis is to measure Hadoop performance in order to show whether Hadoop can be a good solution to store and analyze big data for a small company.

Big data has left its mark on every industry imaginable, including healthcare, advertising, marketing, retail, telecommunications and insurance. In 2017 companies can know more about other businesses and customers than ever before. Data companies compile purchase histories and other vital information and make it available for their client companies which can improve every aspect of their operation in his way. (Gordon 2017.)

Bid gata can help understand customers' preferences better and initiate product improvements. The good news for small companies is that there is already a great deal of source of information and a lot more keeps emerging. In addition, the tools for big data have never been better than they are now. It is about time to start using big data as there is a remarkable quantity of information that can be collected about the customers and the ways to analyze the information (Gordon 2017).

The project starts with installation and configuration of Hadoop without which it would not be possible to perform the test. There are different parameters available to measure the performance. The benchmarking

tools use a well-known meteric called average throughput. The testing chapter explains more about benchmarking in general.

1.3 Research questions

The aim of the thesis is to evaluate Hadoop performance in a virtually distributed environment. The key research questions are:

> *How to analyze Apache Hadoop in pseudo-distributed installation modes?*
> *How to measure the performance in the pseudo-distributed setups?*

The research questions are formulated to grasp the Hadoop setup requirement and the performance measurement perspective. This brings a secondary question into the picture. The subquestions that are derived from the main research questions are:

> *What kind of tools can be used to test Hadoop on an ordinary machine?*
>
> *What kinds of hardware and software are needed to test Hadoop in a normal Microsoft host?*
>
> *What are the configuration parameters for testing a semi-distributed Hadoop environment?*

The installation and configuration chapters as well as the test setup chapter discuss the main research questions.

1.4 Research methodology

The thesis adheres to a research methodology known as design science. Design science research focuses on creation: how things ought to be to attain goals, and to function. (Simon 1996.) Design science research should address either an unsolved problem in a unique and innovative

way or a solved problem in a more effective or efficient way. (Hevner et al. 2004.)

Table 1 shows design science research methodology. The process defines the problem and objective of the solution. It also provides a knowledge base through which the design science research is accomplished.

TABLE 1. Design science research methodology (Geerts 2011)

Activities	Description	Knowledge base
Problem identification	What is the problem?	Evaluate performance of Hadoop cluster. Refer to the research question in Chapter 1 .
Objective of solution	How to solve the problem?	A cluster environment is formed to solve the problem. Refer to Chapter 4.
Design development of artefact	What are the artefacts?	Single node and semi distributed Hadoop installations are required. Refer to Chapter 3.
Demonstration and evaluation	How does the artefact work?	I/O throughput tests are run to evaluate performance. Refer to Chapter 5.
Conclusion	What is the final thought?	See Chapter 6.

The sequences of activities are categorized and placed as phases through which the research methodology is applied,as shown in Figure 1. The phases can be categorized as selection of virtualization environment, installation and configuration of Hadoop in a pseudo-distributed mode, and setting up Hadoop i.e. multi node setup for testing, benchmarking and evaluation phases.

There are a variety of desktop virtualization tools. VMware and VirtualBox stand amongst the most widely used. The tests are carried out on a Lenovo laptop and virtualization software called VirtualBox is used to simulate a cluster environment.

A cluster environment is composed of two or more operating nodes. The thesis starts by installing Hadoop in a pseudo-distributed mode. Upon a successful completion of this mode one or more nodes can easily be added to form a cluster environment. A knowledge base of Hadoop installation in a pseudo and distributed modes is covered in Chapter 3.

In a cluster environment, there is a need for a large amount of RAM memory. Especially the node which will become the master node needs a lot of memory since it holds the metadata of the system. To avoid any negative effect during the test a RAM upgrade to the laptop was made.

The tools used to evaluate the performance can show how well the system operates in each dataset and environment setup. The test is performed by feeding different size data and increasing the number of nodes when needed.

Finally, evaluations and concusions based on the results re presented.

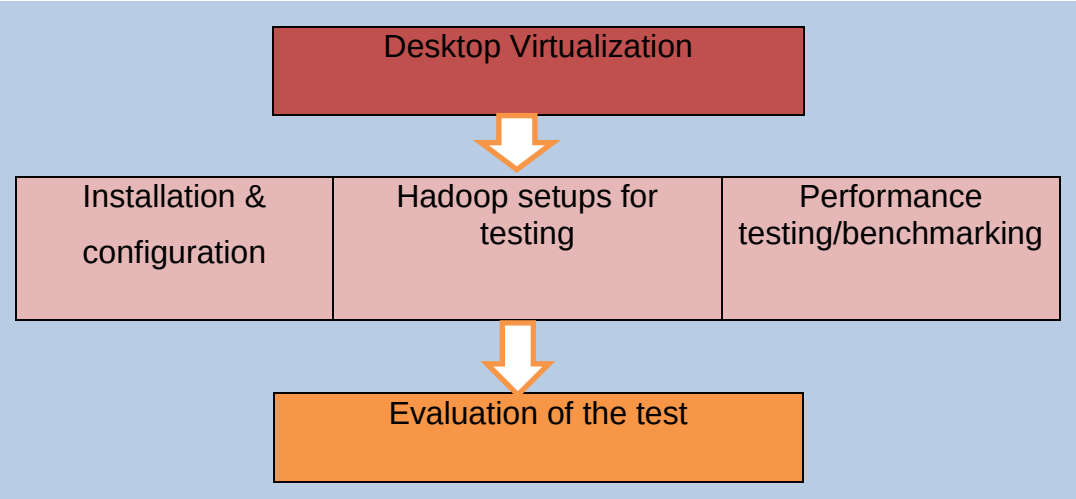

FIGURE 1. Phases of activities

1.5 Structure of the thesis

This document is organized as follows. Chapter 2 introduces and covers the details of the Hadoop ecosystem. Chapter 3 continues with the detail installation and configuration of Hadoop. Chapter 4 discusses the test setups to carry out the performance tests. In Chapter 5, the results obtained are explained. The paper is concluded with Chapter 6 focusing on discussion and analysis of the results obtained and future work is proposed.

2 HADOOP ECOSYSTEM

The Hadoop ecosystem core components (MapReduce and HDFS) are part of Hadoop extended family of proprietary and open source tools. All Hadoop projects are not meant to be used together (Perera 2013).

As shown in Figure 2, the Hadoop ecosystem consists of the core components and other projects that work on top of Hadoop. However, the main concern of the thesis is the data storage and data processing part of the ecosystem.

2.1 Hadoop Distributed File System (HDFS)

HDFS is Hadoop's implementation of a distributed file system. HDFS can hold a very large amount of data, and provides access to this data to many clients distributed across the network. The HDFS concept is represented as master-slave architecture, which has a single NameNode and more than one DataNodes. (Uzunkaya et al.2015.)

2.2 Hadoop projects

The Apache Hadoop projects support several projects for extending Hadoop's capabilities and make it easier to use.

Spark, Pig and Hive are the most common Apache Hadoop projects. Each project is used to create applications to process data in Hadoop. The projects are optimized for specific functions which enables companies to use them for appropriate tasks. (BMC 2016.)

2.2.1 Spark

Spark is both a programming and computing model and provides a gateway for in-memory computing. This has made Spark popular and the most widely adopted Apache project. Spark provides an alternative for MapReduce which enables workloads to execute in memory, instead of in

disk. By implementing in-memory computing, Spark workloads run between 10 and 100 times faster compared to disk execution (BMC 2016).

2.3 Hive

Hive is a data warehousing software that addresses how data is structured and queried in distributed Hadoop clusters. Hive is also a popular development environment that is used to write queries for data in the Hadoop environment. One of the drawbacks of Hive is that it does not support real time queries. (BMC 2016.)

2.4 Pig

Pig is a procedural language for developing parallel processing applications for large datasets in the environment. Pig is another alternative to Java programming for MapReduce.

Pig is widely used for complex use cases that involve multiple data operation. Pig is highly customizable, because users can write their own functions using their preffered scripting language. (BMC 2016.)

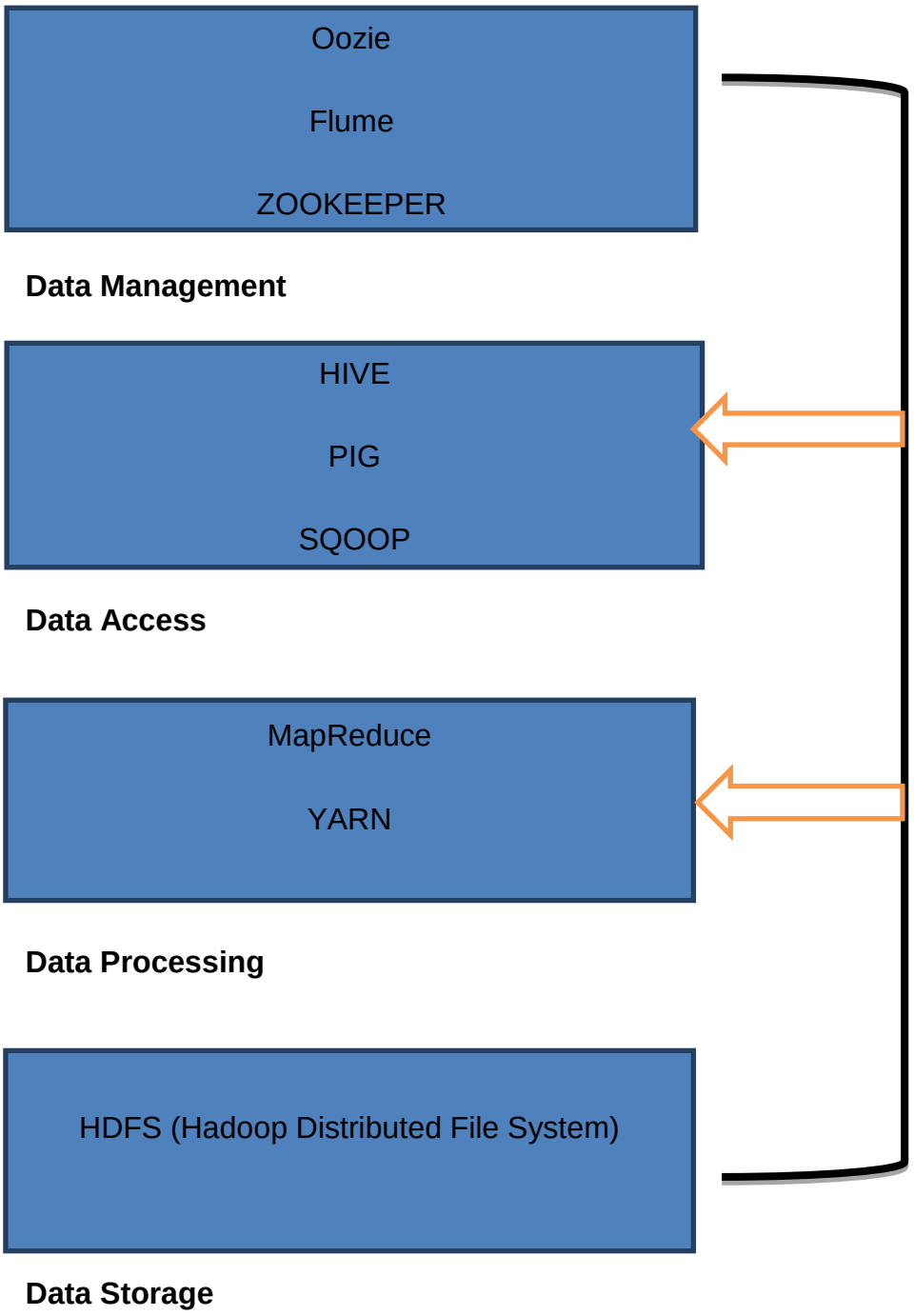

FIGURE 2. Hadoop ecosystem (Korneliusz 2014)

HDFS core properties are data replication, storage capacity and the accessibility of data in a local file system.

2.4.1 Data Replication

HDFS is designed to store very large files across machines in a cluster. It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configaurable per file. HDFS's data replication capability enables storing data reliably, and coping with the malfunctioning or loss of individual machines in the cluster. (Apache software foundation 2014.)

2.4.2 Storage

To be able to store a very large amount of data, HDFS is designed to spread the data across many machines, and to support much larger file sizes compared to distributed filesystems such as NFS.

2.4.3 Data Locality

When a dataset is stored in HDFS, it is divided into blocks and stored across the datanodes in the cluster. When data is not available for the mapper in the same node where it is being executed, the data needs to be copied over the network from the datanode which possesses the data to the datanode which is executing the mapper task. (Hadoop Tutorial 2017.) To better integrate with Hadoop's MapReduce, HDFS allows data to be read and processed locally (Lublinsky et al. 2013). As the performance test is conducted in a distributed environment that involves datanodes spread across the cluster, the data locality feature is crucial for the test.

2.5 HDFS Components

HDFS has a master/slave architecture. An HDFS cluster consists of a single Namenode, a master server that controls the filesystem

namespace. There are also datanodes, one per node in a cluster, which are resposible for storage attached to the nodes that they run on. (Apache software foundation 2014.)

2.5.1 Namenode

The namenode is the core of HDFS system. It is responsible for keeping the directory of all the files in the system. It holds the information where the actual data is stored across the cluster since the namenode itself is not a storage place. The namenode manages the file system and hold all its metadata in RAM. It acts as a file manager on HDFS. The namenode knows the datanodes on which all the blocks for a given file are located.

2.5.2 Datanode

The datanode stores the actual file. It is the most storage intensive component of the HDFS. Datanodes are the worker nodes of the file system. As mentioned in Apache official documentation, datanodes are also responsible for serving read and write requests from the file system’s users. In addition, datanodes perform block creation, deletion and replication upon the instruction from the namenode. Figure 3 shows the overall HDFS architecture.

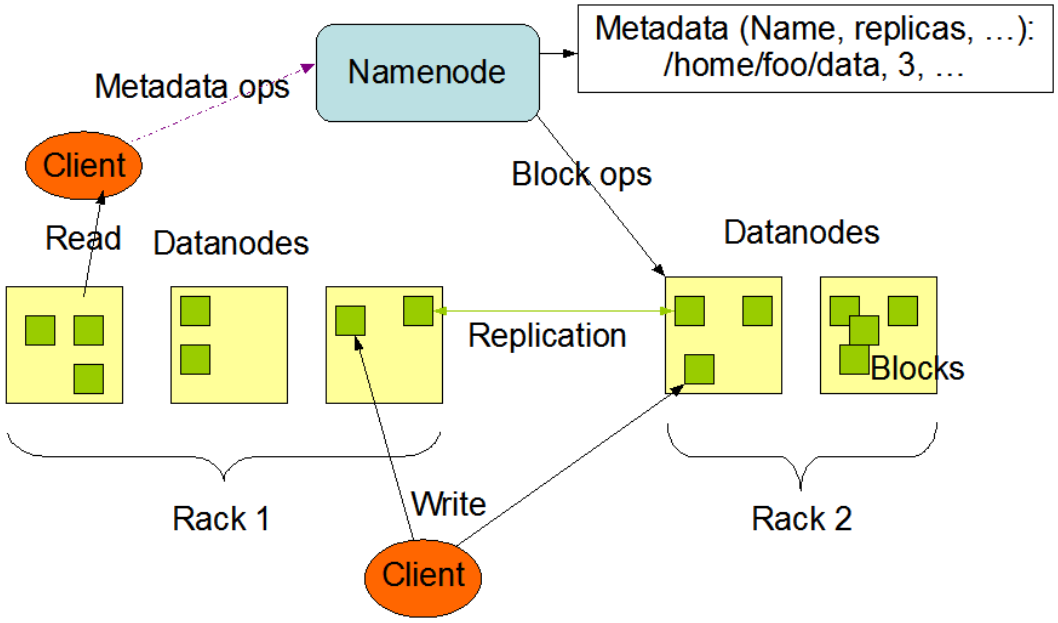

FIGURE 3. Picture of HDFS Architecture (HDFS Architecture)

2.5.3 Secondary NameNode

Secondary namenode is a helper node for the namenode. It can also be regarded as a backup node for the entire HDFS.

2.6 MapReduce

The MapReduce functionality is designed as a tool for deep data analysis, the transformation of very huge amounts of data. It is based on the core concept of parallel programming with high speed. The MapReduce model of programming performs the processing in two distinct phases, which are the map and reduce phases.

Each phase has a key-value pair as input and output The processing rules for the map and reduce functions are defined by the programmer. Hadoop operates the data by dividing the original data into fixed-size pieces called input splits.

One mapper is defined for every split and a key and value pairs are generated by the map function. Each key and values are sorted in mappers. The values are merged when the key is the same.

Then the combiner sums the values for all unique keys for each distributed mapper. This output represents intermediate key value pair and these values are moved to the reduce phase. The reducer has three major phases. These are the shuffle, sort and reduce. (Uzunkaya et al.2015.)

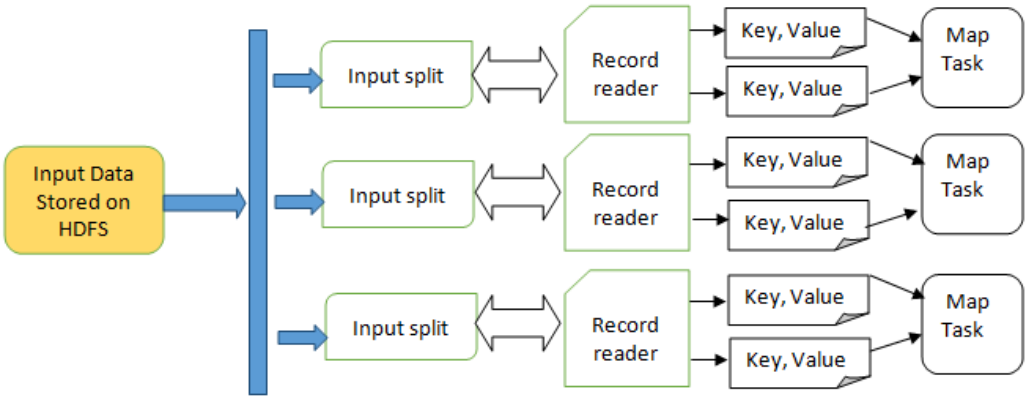

FIGURE 4. MapReduce flow (Abhishek Dharga 2015)

2.7 Hadoop 2.x

Hadoop 2.7.1 is used throughout this thesis. Hadoop version 2 and above are built to overcome the shortcomings of the earlier version Hadoop1. x. In the previous version of Hadoop, resource allocation and job execution was the responsibilities of the JobTracker. However, Hadoop 2.x implements YARN (Yet Another Resource Negotiator).

The primary aim of YARN is to separate issues concerning the resource management and application execution (Karanth 2014).

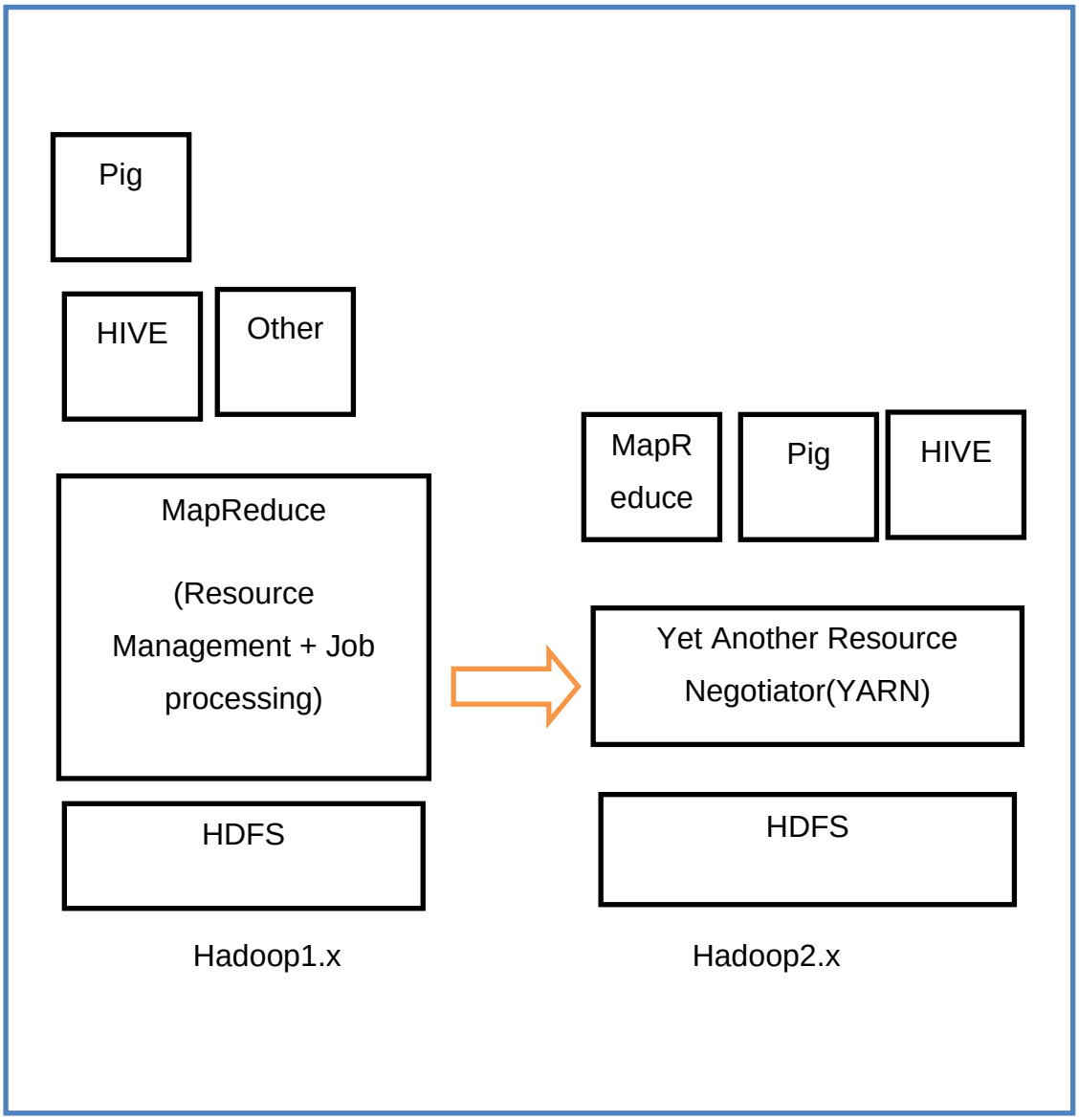

FIGURE 5. Evolution of the YARN architecture

YARN changes the resource management function to a platform layer called Resource Manager (RM). There is a per-cluster RM that monitors the resources and activities of a cluster.

Planning and execution of a job is the responsibility of Application Masters (AM) in the YARN architecture. For instance, there is an AM for each MapReduce job. It must request resource from RM and execute the job (Karanth 2014).

3 HADOOP INSTALLATION AND CONFIGURATION

Hadoop is supported by the Linux/GNU platform and its flavours. It is a prerequisite to install the Linux operating system for setting up the Hadoop environment (Hadoop Tutorials 2017.) As pointed out earlier, the test is conducted in a Windows host machine, which is why desktop virtualization software called VirtualBox is used and CentOS, one of the Linux flavours, is installed.

Hadoop can be installed in many different modes. The Hadoop modes determine how the different Hadoop components execute.

3.1 Local Standalone mode

This mode is Hadoop's default mode. The Standalone mode requires no configuration and all Hadoop components, such as Namenode, Datanode, JobTracker and TaskTracker, run in a single Java process (Turkington 2013). The local standalone is not covered in the thesis.

3.2 Pseudo-Distributed mode

In this mode, a separate JVM is spawned across the components of Hadoop. It can be considered a minicluster setup on a single host (Turkington 2013). The pseudo-distributed mode is used as a stepping stone to carry out the *fully* distributed setup and further in testing.

3.3 Fully distributed mode

In this mode, Hadoop is spread across multiple machines in the cluster. Even though it requires a lot of configuration work, the fully distributed mode is the one that can scale Hadoop across a cluster of machines (Turkington 2013).

3.4 Pseudo-Distributed mode installation and configuration

In the pseudo-distributed mode, a single node acts as a master node and slave node. It is equivalent to simulating a distributed environment on a single server. In addition to this, it acts as master JobTracker and slave TaskTracker.

The pseudo-distributed mode helps to familiarize with the Hadoop configuration. A successful installation and configuration in this mode paves the way for the upcoming multi-node cluster setup. The following steps are a prerequisite to install Hadoop in the pseudo-distributed mode (Prajapati 2013).

3.4.1 Java Installation

Hadoop needs Java regardless of which type of Hadoop installation is used. Hadoop requires Java version 1.5 or above. The screenshot from the node assures whether Java is sucessfuly installed or not as shown in Figure 6.

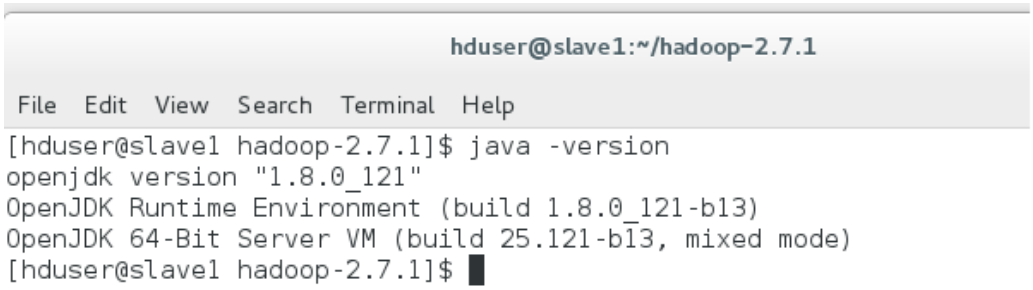

FIGURE 6. Java installation

If the above command generates no output, a full Java install is required. After Java is properly installed, the Java environmental variable is set in the *bashrc* file as shown in Figure 7.

```
# User specific aliases and functions
export HADOOP_HOME=$HOME/hadoop-2.7.1
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.1/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.1
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.1
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.1
export YARN_HOME=$HOME/hadoop-2.7.1
export PATH=$PATH:$HOME/hadoop-2.7.1/bin

#set JAVA_HOME
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.121-0.b13.el7_3.x86_64/j
e
export PATH=$JAVA_HOME/bin:$PATH
".bashrc" 22L, 663C                                          22,1          Bot
```

FIGURE 7. Java environment variables

The commands enable Hadoop to check that the correct version of Java is installed and available from the command line without having to use the lengthy path names to refer to the install location. These setups will be lost after logging out. Adding the files to the *bashprofile* make the configuration always available at start up (Guo 2013).

3.4.2 Hadoop user

There are different system users in any Linux-based systems. Identifying the user that is eligible to run Hadoop is a crucial step. The root has a privilege to create the user as shown in Figure 8.

```
le  Edit  View  Search  Terminal  Help
duser@slave1 ~]$ sudo su
udo] password for hduser:
RT has detected 1 problem(s). For more info run: abrt-cli list --since 1488233
2
oot@slave1 hduser]# useradd fitsum
oot@slave1 hduser]#
```

FIGURE 8. Hadoop user creation

3.4.3 SSH

Hadoop requires communication between multiple processes on one or more machines. Because of that, the Secure Shell (SSH) key pair that has

an empty pass phrase is needed to ensure that the Hadoop users can connect to each required host without needing a password (Turkington 2013).

After generating the key pair, the new public key is added to the list of trusted keys. The public key becomes trustworthy when Hadoop is trying to connect.

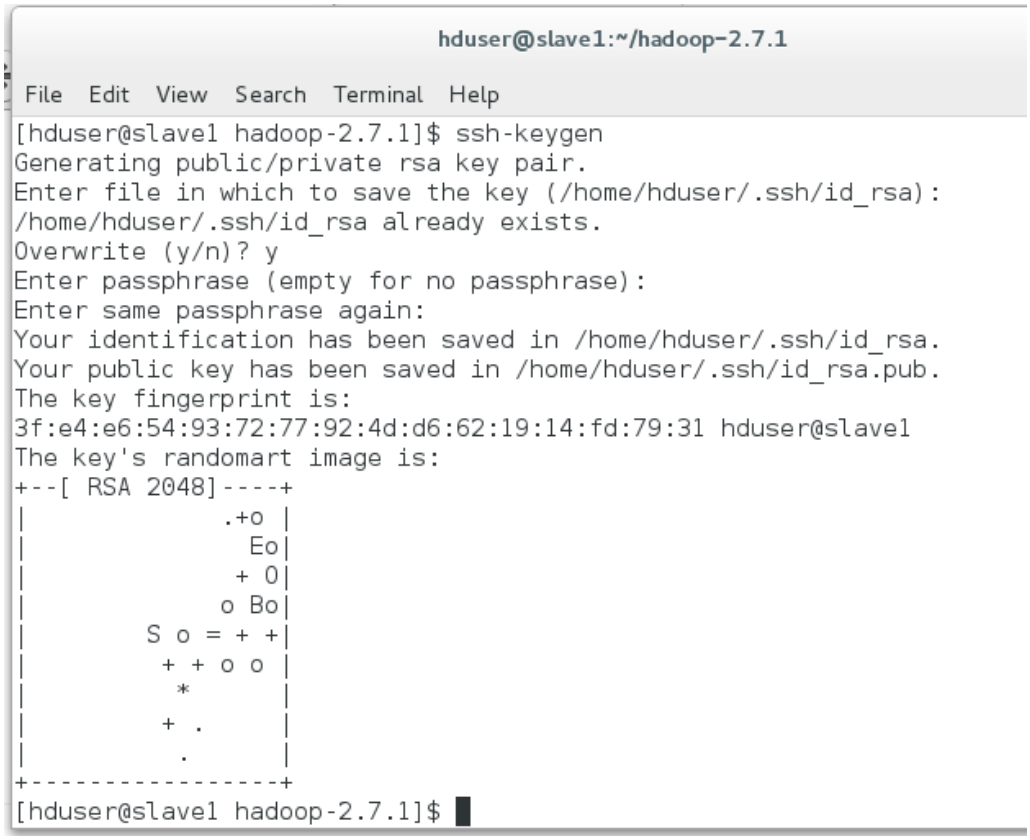

FIGURE 9. SSH key generationg

3.5 Hadoop Installation

The Hadoop version 2.7.1 from the Apache website is downloaded and unpacked inside the Hadoop user created as shown in Figure 10.

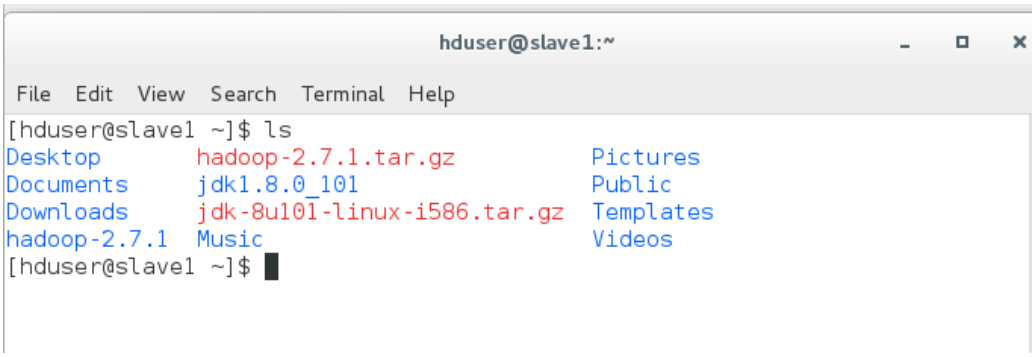

FIGURE 10. Unextracted Hadoop version 2.7.1

3.5.1 Hadoop environmental variables

Hadoop environmental variables need to be set the same way as Java environmental variables, as shown in Figure 7. All the files such as the MAPRED, YARN, HDFS use this short path to reference. The file can be appended by using a vim or any other editor (Turkington 2013).

Figure 11 shows the command that keeps the change every time the bash file is edited.

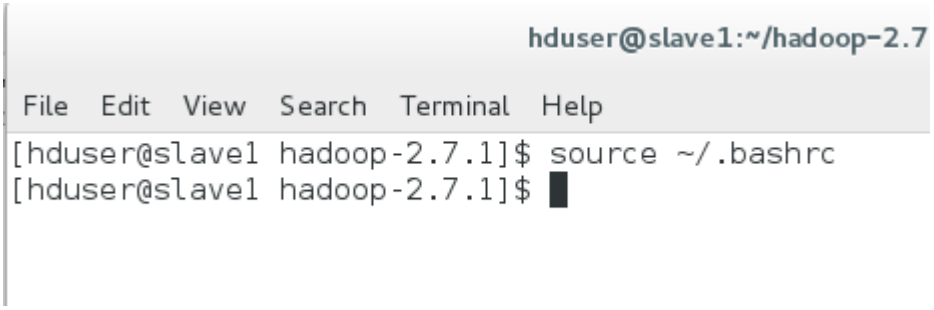

FIGURE 11. Sourcing the bash file

3.5.2 Hadoop configuration files

In addition to the environmental variables, there are a few more configuration files that need to be configured so that the Hadoop single node setup works without any problem. These files are *HDFS-site.xml, yarn-site.xml, mapred-site.xml, Core-site.xml* and the *Hadoop-env.sh.*

3.5.3 The HDFS-site.xml

This file contains information such as value of the replication data, namenode path and datanode path of the file system. It is the place where Hadoop infrastructure is stored. The *dfs. replication* variable specifies how many times replication is carried out in each HDFS block (Mar 2015). See Appendix 1.

3.5.4 The Hadoop-env.sh

This file is intended to hold the location of Java. It helps the Hadoop operation get hold of the Java as shown in Figure 7.

3.5.5 The core-site.xml

The core-site.xml informs Hadoop daemons where the namenode runs in the cluster (edureka! 2017). The Hadoop daemons are Namenode, Secondary Namenode, Jobtracker, Datanode and Tasktracker. This file contains key-value pairs. For example, the *fs.default.name* variable holds the location of the namenode. See Appendix 2.

3.5.6 The mapred-site.xml

“The mapred-site.xml contains configuration settings of MapReduce such as th number of JVM that can run in parallel, size of mapper and CPU core available for process” (edureka!2017). See Appendix 3.

3.5.7 The yarn-site.xml-

The default value serves well for this purpose. The file contains configuration information that overrides the default values of core properties stored in the YARN parameter file (Apache software foundation 2014).

3.6 Namenode setup

Formatting enables the creation of an empty file system by creating the storage directories and the initial versions of the namenode’s persistent data structure (Perera & Gunarathne 2013).

The last step is to start the HDFS. The daemons, namely the namenode, the datanode and the secondary namenode, will start on the local machine. The daemons can be viewed in the web to check whether they are installed successfully or not (Lublinsky et al. 2013).

4 TEST SETUPS

A significant amount of time has been devoted to the actual configuration of Hadoop due to lack of official documentation for Hadoop installation in VirtualBox, which has different flavours of Linux inside. There were numerous failures due to instability. The main challenges are listed down at the end of this chapter. Hence, configuration takes up a vast part of the chapters.

The whole idea is to deploy a multi-node cluster on two nodes. The nodes act as masters in their standalone status. However, in the new setups one acts as a master the other as a slave. The master, however, serves as slave to process datanode daemons.

Figure 12 demonstrates the overall approach of the project. Two stand-alone machines form a multi-node cluster. The configuration details below show the ip address of each node (masters and slaves) and the services they run.

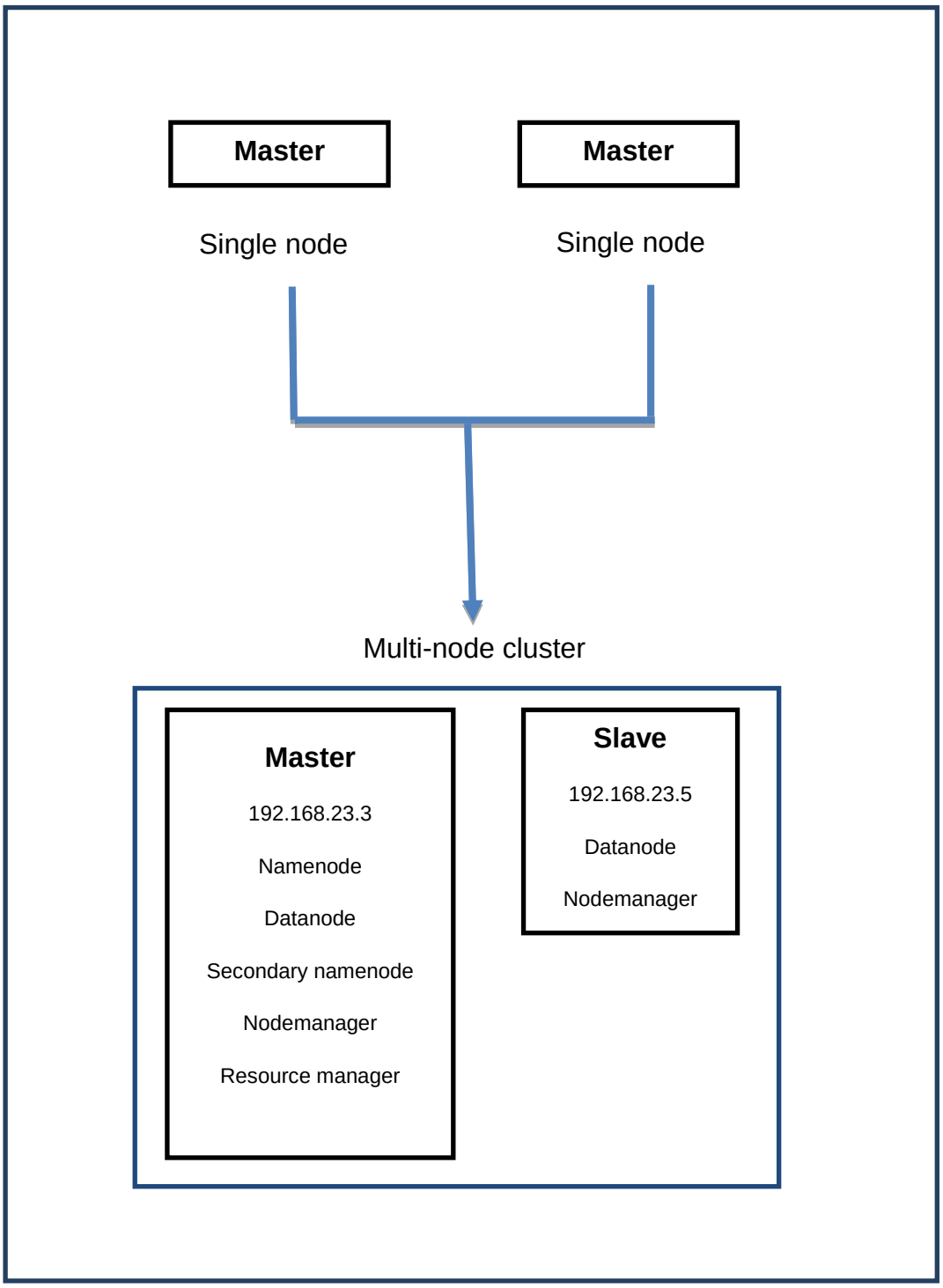

FIGURE 12. Cluster topology evolution

4.1 Cloning the base virtual machine

Clonning is a process of replicating the virtual machine into several identical machines with the same configuration setups. It minimizes the task of creating new virual machines whenever a need for a node arises. However, custom configurations are necessary. For example, the master node requires more RAM than the slave nodes. Unlike the masters, the slaves require storage space. All slave nodes are replicas of the original virtual machine. Figure 13 displays the look of virtual machines running in VirtualBox (VirtualBox 2017).

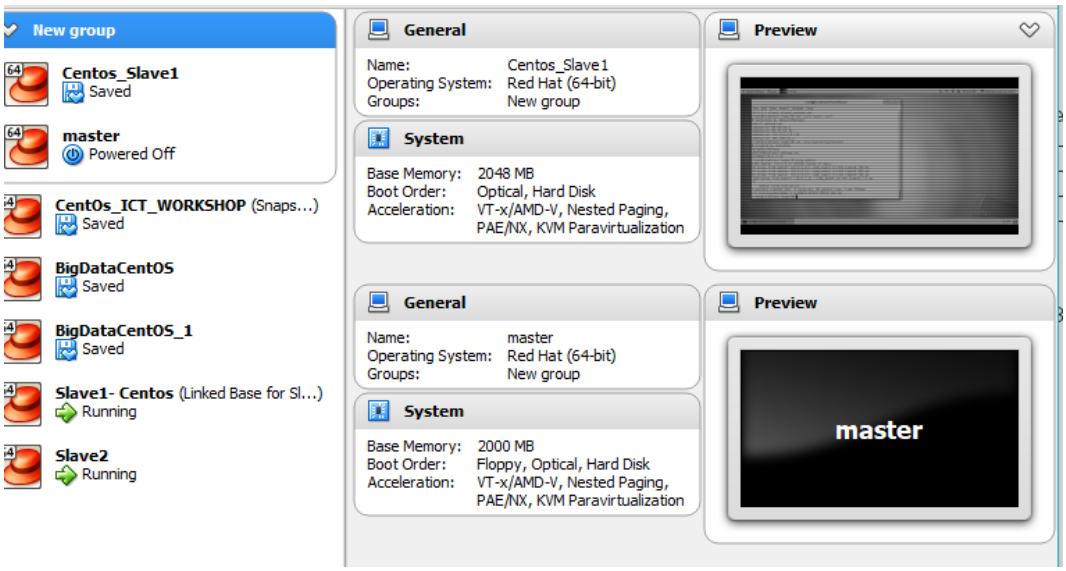

FIGURE 13. Clone of virtual machine

4.2 Networking

The nodes in the cluster should form a network to be able to communicate with each other. To form a successful network, the below mentioned files are updated in each of the machines involved (edureka 2017).

4.2.1 The *etc/hosts* file

Each node in the cluster updates this file in order to form a network. Figure 14 shows how each of the machines updates the file.

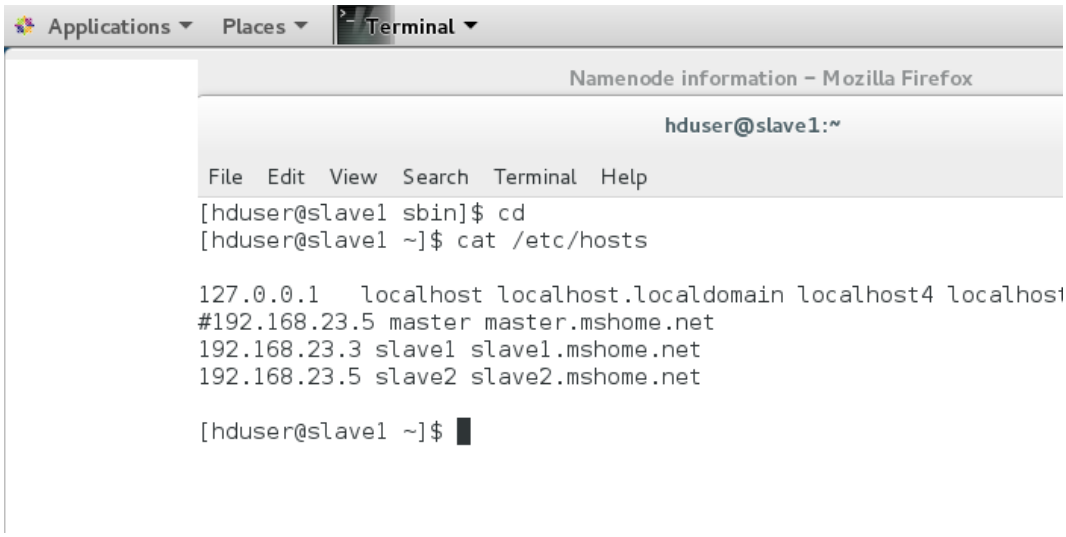

FIGURE 14. Hostname and ip address information

4.2.2 The etc/sysconfig/network

The *etc/sysconfig/network* file is used to specify information about the desired network configuration (CentOS Deployment Guide 2008).

4.2.3 The interface

The *etc/sysconfig/network-scripts/ifcfg-enps03* file contains files related to the network interface. The most important properties are the BOOTPROTO and the ip address. The BOOTPROTO value must be static with the corresponding ip information as shown in Figure 15. The Dynamic ip address scheme is not good since it allocates the ip address periodically and the nodes might malfunction otherwise (CentOS Deployment Guide 2008.)

With the absence of official documentation on how to configure this type of cluster, getting the network ready was challenging. Due to this, it is important to explain networking.

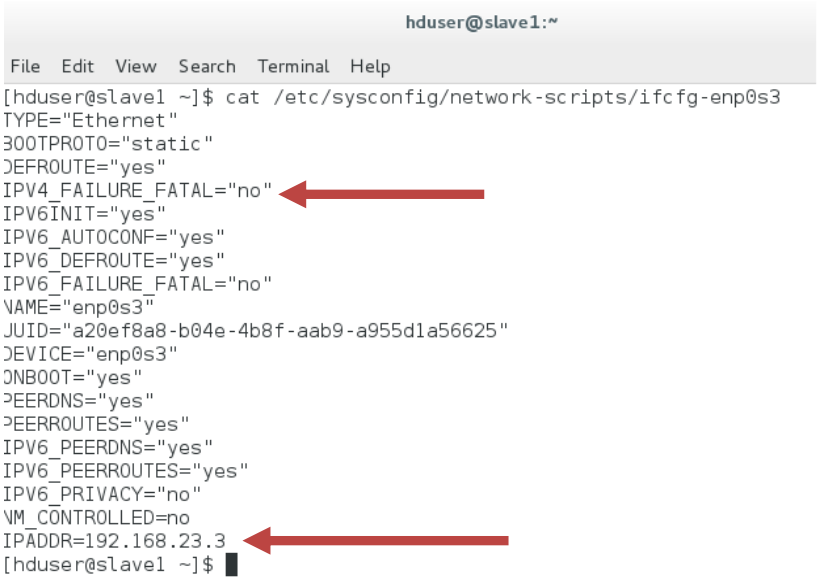

FIGURE 15. The network interface information

4.3 Installation and configuration

Most of the installation, for example Java and Hadoop, and configurations are already peformed in the pseudo-distributed mode. This section covers the installation and configuration parts that are important to perform the performance tests.

4.3.1 The HDFS-site.xml

The only addition when forming a cluster is to add a directory to hold the namenode and datanode directories. Directories must be physically formed and the xml file must have this information. The master node holds the namenode and datanode directories , so it also serves as a slave, whereas the slave nodes hold datanode directories only. Figure 16 shows the master *HDFS-site.xml* file with the namenode and datanode directories.

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/home/hduser/hadoop-2.7.1/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hduser/hadoop-2.7.1/hdfs/datanode</value>
</property>
</configuration>
[hduser@slave1 hadoop]$
```

FIGURE 16. HDFS-site.xml file

4.3.2 The core-site.xml

The only addition from the previous mode setup is to inform the nodes about the server that acts as a master. Therefore, the fs.default.name holds the ip address or hostname of the master. The nodes in the cluster update the respective file as shown in Figure 17 (Lublinsky et al.2013).

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://slave1:8020</value>
</property>
</configuration>
[hduser@slave1 hadoop]$
```

FIGURE 17. Core-site.xml file

The *yarn-site.xml* and *mapred-site.xml* files are kept without much change from previous configuration in Chapter 3.

4.3.3 The slaves

The slave file which is included inside the Hadoop configuration file serves a very important task. While maintaining a cluster that requires a master-slave architecture, nodes need to know which are masters and which are slaves. Accordingly, the slave file in the respective nodes holds the information of master (edureka 2017).

For example, the master node in both the master and the slave file holds th ip address or hostname of the node which acts as a master. The slave file in the master node holds the hostnames or ip addresses of all slave machines. This is very important inorder to know where to run datanode services and namenode services (Lublinsky et al. 2013).

4.4 Namenode setup

Formatting the namenode takes place at the master node. After a successful format, each Hadoop and yarn service starts in the master node.

The slave node automatically gets datanode and nodemanager daemons started as shown in Figure 19. The format and starting service commands can be found below respectively (Lublinsky et al. 2013).

The Jps command is used to check all the Hadoop daemons are running on the machine. Figures 18 and 19 display what daemons each node is running in the master and slave node respectively. It must be noted that slave 1 has become the new master in the cluster formation.

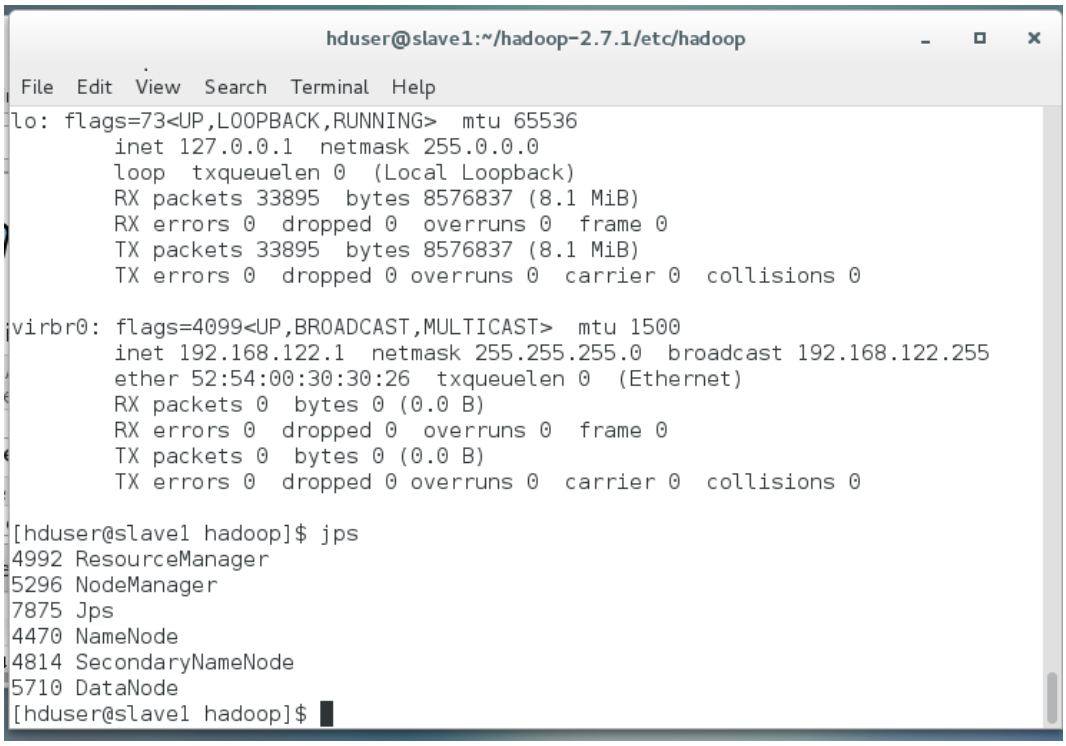

FIGURE 18. Jps command from master node

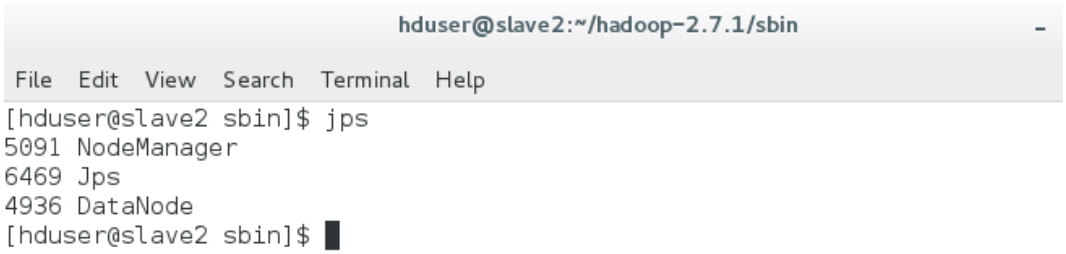

FIGURE 19. Jps command from slave node

4.4.1 Web UI

The namenode and datanode health can be seen from the web. As shown in Figure 20, there are two live nodes, which is a good sign of a semi-distributed cluster setup (Noll 2017.)

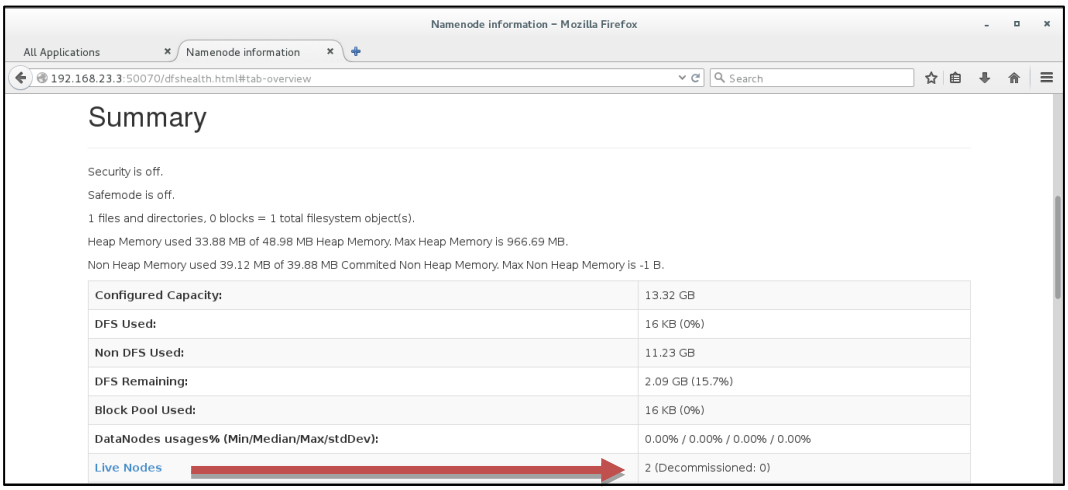

FIGURE 20. WebUI namenode

4.4.2 Problems

There were many problems regarding this kind of test due to lack of documentation at the installation and configuration stages. There were limited resources regarding VirtualBox and Hadoop performance measurement experiments. Networking amongs the nodes failed frequently. This is one reason network configurations are covered well so that future work based on the thesis becomes easier. As shown in Figure 15, the problem was solved by assigning a static ip address instead of a dynamic one.

In addition, datanodes in the master and slave nodes encountered difficulties to run. This was overcome by removing the content of the directory before starting Hadoop services using the command rm -f.

5 PERFORMANCE TEST AND RESULTS

5.1 TestDFSIO

Hadoop is equipped with inbuilt test jar files. The benchmarking or performance testing jar files are among the files.

TestDFSIO tests the I/O performance of HDFS. Using the MapReduce job is a convenient way to read or write files in parallel. Each file is read or written in a separate map task, and the output of the map is used for collecting statistics relating to the file just processed. The statistics are stored in reduce to produce a summary (White 2015.)

Throughput performance measure is the main metric for testing the cluster. The reason for choosing this benchmark is that no real data is required to perform the test. This test has been used widely and proved to be useful in determining performance blockage in the network and conducting stress tests on HDFS. The test proved to be highly useful and economical in terms of hardware and setup costs and to show the first impression of how fast a cluster performs in terms of input output (Chaudhary & Singh 2014). The program generates the required number of files for testing. With the scarce resources available TestDFSIO is a better choice.

The next chapters deal with the testing preparations and evaluation of the test results.

5.2 Testing preparations

The testing was done on a single-node and multi-node cluster with the same hardware and software specifications. This enables to make a reasonable evaluation. RAM memory 2GB and Hard Drive 8GB is for each master or slave regardless of the mode of installation.

The testing mainly focused on measuring a performance metric called write throughput. It is the average time needed to write or read all files.

5.3 TestDFSIO write

The benchmark program *TestDFSIO.java* uses the MapReduce concept for writing/reading files. It sets map tasks equal to the value of *-nrFiles* and each map task will write/read the data on a datanode completely. After each map task is done, it collects the values of tasks (number of map tasks), size (filesize)and time (executing time)and sends them to the Reducer. The Reducer counts all the immediate values and saves a reduced output file named *"part-00000"* on the HDFS. Using this file, the program computes data throughput, average I/O rate and I/O rate standard deviation (Tien Duc Dinh 2009.)

The first test was carried out by varying the number of files and recording the throughput, average I/O rate, I/O standard deviation and test execution time. Figure 21 shows the actual test result as seen from the node.

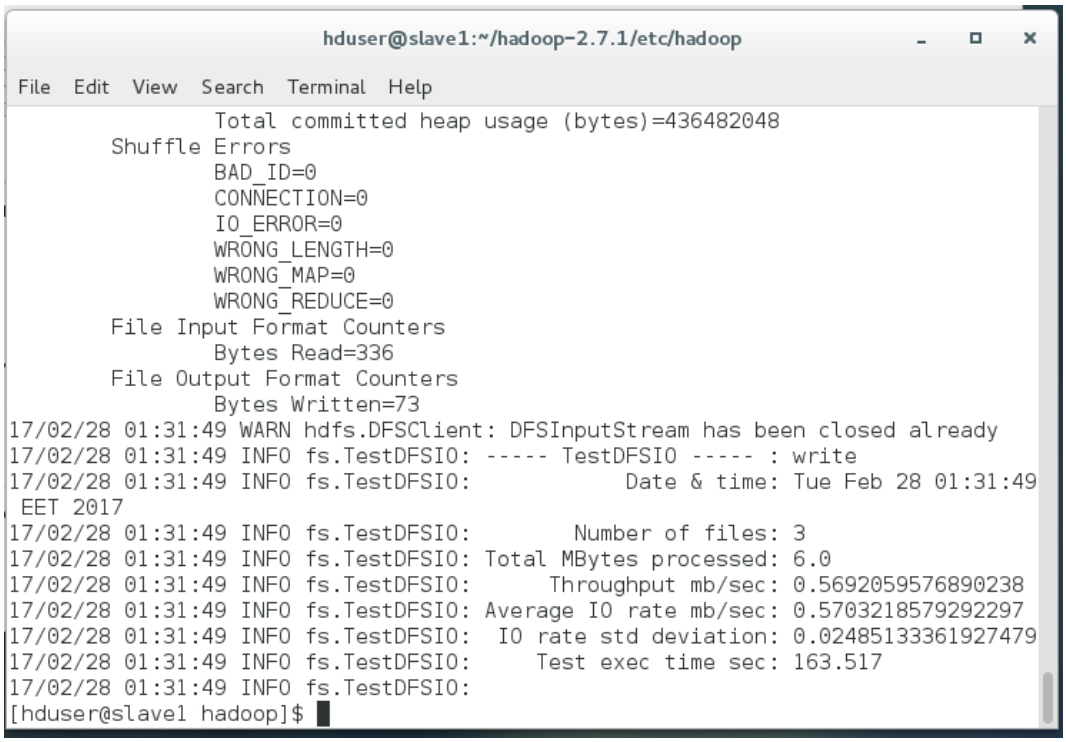

FIGURE 21. Actual test results

Tables 2 and 3 show the results of the tests on single node and multi-node cluster respectively.

TABLE 2. Actual results of multi-node write tests

Number of files	Mega bytes processed	Throughput mb/sec	Average I/O rate	I/O rate standard deviation	Test execution time(sec)
1	10	5.14668039114771	5.1466803550720215	8.483444978942536	80.94
2	20	2.591344908007256	2.591378927230835	0.00944426294634878	99.313
3	30	1.6279574560451486	1.6282248497009277	0.02092147274894543	118.811
4	40	0.7020746332257	0.7374733686447144	0.14256587105985868	205.987

TABLE 3. Actual results of single-node write tests

Number of files	Mega bytes processed	Throughput mb/sec	Average I/O rate	I/O rate standard deviation	Test execution time(sec)
1	10	6.176652254478073	6.176652431488037	2.2233040652974844	105.143
2	20	2.459419576979833	2.4594316482543945	0.00546027568631941	130.325
3	30	0.3801510466825485	0.38442814350128174	0.04208925259505002	189.693
4	40	0.3159482792666840	0.32743513584136963	0.05505713638494824	224.666

In addition to the statistics shown in the tables above, the web GUI displays the number of live datanodes in operation. This makes it convenient to monitor any missing nodes while running the test. Both nodes are in service as shown in Figure 22.

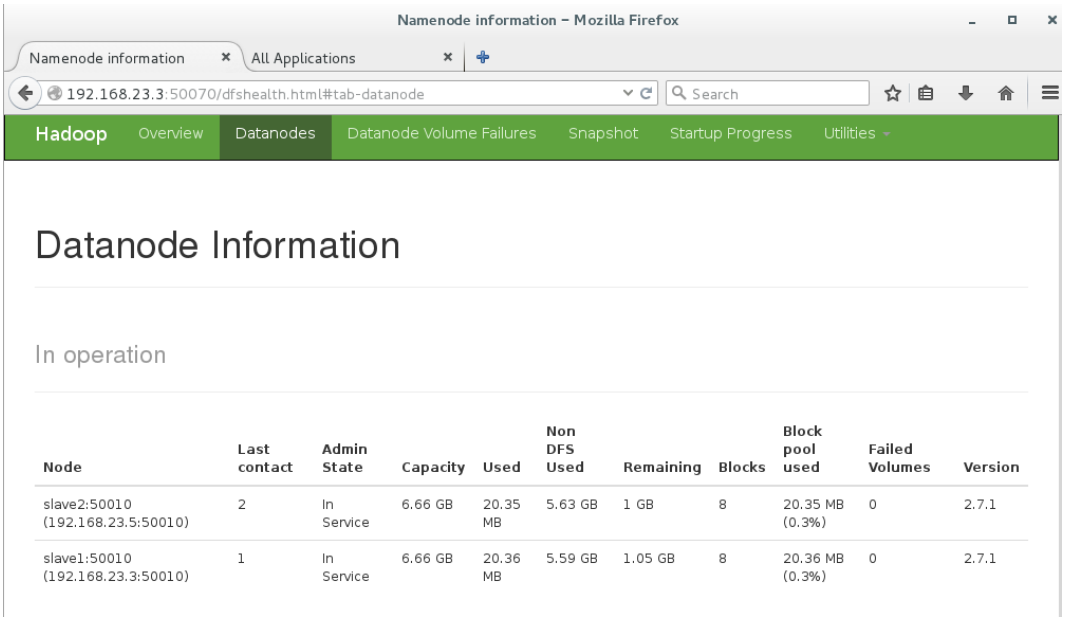

FIGURE 22. Datanode information

5.4 TestDFSIO read

TestDFSIO read was performed with the same fashion as the write tests. The read test of the TestDFSIO does not generate its own input files. For this reason, the read test uses the generated output files from the write test. See Table 4.

TABLE 4. Read test results

Number of files	Mega bytes processed	Throughput mb/sec	Average I/O rate	I/O rate standard deviation	Test execution time(sec)
1	10	27.7777777777778	27.7777786254828	0.00221311627068342	76.059
2	20	39.21568627450981	39.334245140380859	2.1594499575514994	88.932
3	30	25.380710659898476	25.968849182128906	3.7140825603664003	107.357
4	40	10.576414595452142	10.0952651964994468	0.0952651964944689	123.564

6 CONCLUSIONS AND DISCUSSION

The main aim of the thesis was to evaluate Hadoop performance on a well-known metric called throughput and reveal the necessity for small companies to consider Hadoop in this data centric business era. The TestDFSIO test was performed in order to assess Hadoop I/O performance of single node and multi node Hadoop configurations. The test was run increasing the size of the file from one to four with size 10MB each.

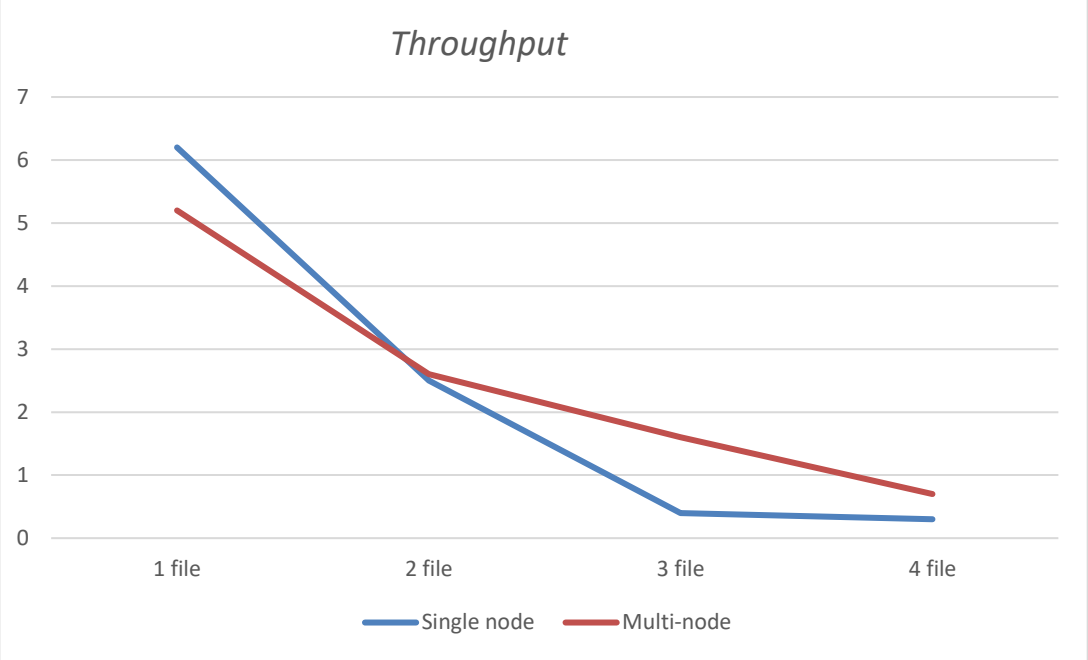

FIGURE 23 TestDFSIO write performance summary

As shown in Figure 23, the multi node cluster has better performance than a single node when it deals with a few files at a time. However, the performance of a multi node cluster decreases as the number of files and data use increases. There are different reasons for this, among which network traffic and local disc access could be mentioned. The execution time is also 20-30% faster than that of a single-node as shown in Table 3.

The read throughput performance is analogous to the write performance test results. It further strengthens the write throughput performance evaluations (see Figure 22).

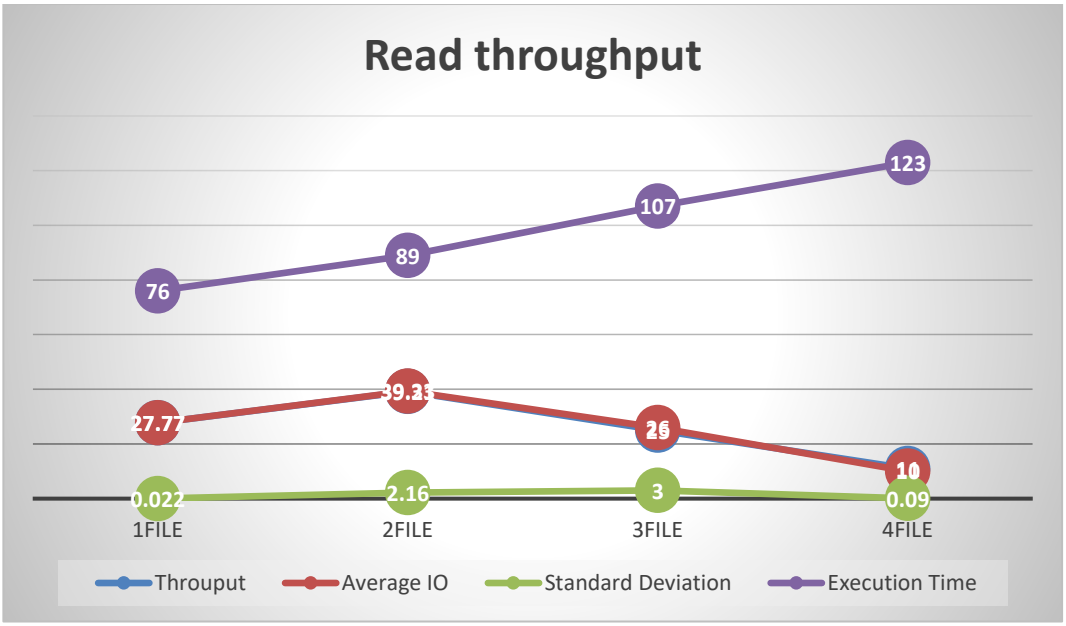

FIGURE 24. Read throughput

It is evidently possible to conclude that with inexpensive infrastructure and by optimizing existing resources it is possible to process large volumes of data. Even though this thesis does little about virtualized Hadoop, the test can pave a way for wider research. As the results show, virtualized Hadoop can be a good alternative for performance reasons too.

6.1 Research questions

There were several research questionthat needed answers during the development. The chapters that followed the research question answers all the questions.

How to analyze Apache Hadoop in pseudo-distributed installation modes?

What kind of tools can be used to test Hadoop on an ordinary machine?

How to measure the performance in the pseudo-distributed setups?

Chapters 1 and 3 of the thesis discuss the first two questions. Not only the the questions, but also the Hadoop installation and configuration are discussed in detail. As the aim, from the outset, was to operate on a single host machine, it was not a simple problem to tackle. This is because books as well as other online documentation focus mainly on a fully distributed environment where there are thousands of nodes for operation. However, there was a chance to carry out the investigation using a virtually distributed environment. As stated in the Chapter 3, VirtualBox was successfully used in the thesis

Understanding how to accomplish the test was tip of the iceberg. The core aim of the thesis still needed to be addressed. In addition to that, the thesis needed to answer how to form a cluster environment to perform the Hadoop performance test. Chapters 4 and 5 deal with forming cluster environment and selection of input-output(I/O) performance measure respectively.

The following subquestions raised in the research question are dealt with in subsquent chapters of the thesis and incorporated to the main research questions. Even though the test was carried out in VirtualBox, it is possible to do the test in any available hardware. To perform performance testing in a semi or fully distributed environment, there has to be a network of nodes. The nodes must stay connected to perform Hadoop related tasks. Chapter 4 deals with all networking requirements as well as the presumed topology to carry out the test.

What kinds of hardware and software are needed to test Hadoop in a normal Microsoft host?

What are the configuration parameters for testing a semi-distributed Hadoop environment?

6.2 Limitations and future work

The performance evaluation was conducted in a virtualized environment as mentioned in earlier chapters. The results can show a general overview of Hadoop performance for small companies to have the courage to try it out. Due to lack of real hardware for testing, the network and hardware factors that might degrade the performance of Hadoop were not measured. This thesis can be more enhanced and a better result can be achieved with more physical nodes added to the cluster.

REFERENCES

Abhishek D. 2015. Understanding map-reduce programming.

Apache Software Foundation. 2014. Welcome to Apache™ Hadoop®! [accessed 3rd October 2017]. Available in: http://hadoop.apache.org/

BMC. 2016. Hadoop Ecosystem and Components [accessed 3rd October 2017]. Available in: http://www.bmc.com/guides/hadoop-ecosystem.html

CentOS Deployment Guide. 2008 [accessed 3rd October 2017]. Available in: https://www.centos.org/docs/5/html/5.2/Deployment_Guide/

Chaudhary, U. & Singh, H. 2014. MapReduce Performance Evaluation through Benchmarking and Stress Testing On Multi-Node Hadoop Cluster. International Journal of Computational Engineering Research (IJCER), Vol 04, Issue 5, p. 2250 - 3005. [accessed 3rd October 2017]. Available in: http://www.ijceronline.com/papers/Vol4_issue05/version-2/H04502048051.pdf

Eadline, D. 2015. Running MapReduce Example Programs and Benchmarks. informIT, Pearson. [accessed 3rd October 2017]. Available in: http://www.informit.com/articles/article.aspx?p=2453563

Edureka! 2017. Edureka Hadoop tutorial [accessed 3rd October 2017]. Availablein: https://www.youtube.com/watch?v=tu7nCsHImbI&list=PL9ooVrP1hQOFrYxqxb0NJCdCABPZNo0pD

Elgendy N. & Elragal A. 2014. Big Data Analytics: A Literature Review Paper. In: Perner P. (eds) Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2014. Lecture Notes in Computer Science, vol 8557. Springer, Cham

Geerts, G. L. 2011. A design science research methodology and its application to accounting information systems research. International

Journal of Accounting Information Systems, Vol 12, Issue 2, p. 142-151. Pergamon.

Gordon, A. 2017. How Your Small Business Can Make Use Of Big Data Now. Forbes. [accessed 3rd October 2017]. Available in: https://www.forbes.com/sites/forbestechcouncil/2017/05/26/how-your-small-business-can-make-use-of-big-data-now/#13d86afd2dc6

Guo, S. 2013. Hadoop operations and cluster management cookbook. Packt Publishing.

Hadoop Tutorial. 2017. Learn Hadoop, Simply Easy Learning. Tutorials Point [accessed 3rd October 2017]. Available in: https://www.tutorialspoint.com/hadoop/

Hevner AR, March St, Park J, Ram S. Design Science in information systems research. MIS Q 2004.

Hortonworks. 2017. Apache Hadoop Overview [accessed 3rd October 2017]. Available in: https://hortonworks.com/apache/hadoop/

Karanth, S. 2014. Mastering Hadoop. Packt Publishing.

Korneliusz. 2014 Hadoop Ecosystem and BigData

Lublinsky, B., Smith, K. T., and Yakubovich, A. 2013. Professional Hadoop Solutions. John Wiley & Sons

Marr, B. 2015. Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance. John Wiley & Sons.

Murthy, A. C., Vavilapalli, V. K., Eadline, D., Niemiec, J., & Markham, J. 2013. Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2. Pearson Education.

Noll, M. G. 2017. Running Hadoop on Ubuntu Linux (Multi-Node Cluster) [accessed 3rd October 2017]. Available in: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster

Perera, S. & Gunarathne, T. 2013. Hadoop MapReduce Cookbook. Packt Publishing.

Prajapati, V. 2013. Big Data Analytics with R and Hadoop. Packt Publishing.

Simon HA, The science of the artificial. 3rd ed. Cambridge, MA: MIT Press:1996

Tien Duc Dinh. 2009. Hadoop Performance Evaluation [accessed 3rd October 2017]. Available in: https://wr.informatik.uni-hamburg.de/_media/research/labs/2009/2009-12-tien_duc_dinh-evaluierung_von_hadoop-report.pdf

TechTarget. 2006. relational database [accessed 3rd October 2017]. Available in: http://searchsqlserver.techtarget.com/definition/relational-database

Turkington, G. 2013. Hadoop Beginner's Guide. Packt Publishing.

Uzunkaya, C., Ensari, T. & Kavurucu, Y. 2015. Hadoop Ecosystem and Its Analysis on Tweets. Procedia-Social and Behavioral Sciences, 195:1890–1897.

VirtualBox 2015 documentation] Available in: https://www.virtualbox.org/manual/ch01.html

White, T. 2015. Hadoop: The Definitive Guide, 4th Edition. O'Reilly Media.

APPENDIX 1

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namendode.name.dir</name>

<value>/home/hduser/Hadoop-2.7.1/HDFS/namenode</value>

</property>

</configuration>

APPENDIX 2

<configuration>

<property>

<name>fs.default.name</name>

<value>HDFS://localhost:8020</value>

</property>

</configuration>

APPENDIX 3

<configuration>

<property>

<name>MapReduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>